Estd : 1984	**KONGU ENGINEERING COLLEGE** **(Autonomous)** **PERUNDURAI – 638060** **INTERNAL QUALITY ASSURANCE CELL** **Laboratory Manual**	TÜV SÜD ISO 9001

Department of Information Technology.

Prepared By

V.P.Gayathri
S.VinothKumar

Approved By

Dr.S.AnandhaMurugan
HoD/IT

LIST OF EXPERIMENTS

KONGU ENGINEERING COLLEGE
(Autonomous)
PERUNDURAI – 638060
INTERNAL QUALITY ASSURANCE CELL

Laboratory Assessment Rubrics

Department of Information Technology

Academic Year: 2024-2025 **Semester: Even**
Course code & Name: 20ITL62 – Internet of Things Laboratory **Class / Sec: III IT A&B**

Criteria/Marks assigned	**Good**	**Satisfactory**	**Need to improve**
Conduct of experiment(25)			
Ability to understand and analyze the circuit design.(10)	Ability to understand and Implemented the circuit successfully and get the output (10)	Partially implemented the circuit (9-6)	Demonstrates minimal or no ability to articulate the design of circuit(5-1)
Identify a suitable circuit design and apply for the problem domain (5)	Can able to design the circuit and apply for the problem domain (5)	Designed the circuit but unable to apply for the problem domain (4-3)	Have less knowledge in identifying the technique (2-1)
Ability to use specific programming constructs for implementing the circuit.(10)	Program logic for implementing circuit is correct with no known boundary error and no redundant and contradictory condition(10)	Program logic is correct ,but may contain boundary error and redundant and contradictory condition(9-6)	Program logic is not appropriate for implementing the circuit(5-1)
Observation and record(25)			
Interpretation of	Successfully implemented the circuit and interpretation of findings a	Provides interpretation of the	Minimal interpretation of the findings(5-1)

findings(10)	correctly solve the problem(10)	findings but does not derive the logical conclusion of the problem(9-6)	
Correctness(5)	Program construct produce correct output as expected(5)	The program construct produces partial output(4-3)	The program does not produce correct output.(2-1)
Adherence experiment completion(10)	On time completion of the experiment(10)	Completion on next day(9-6)	Delayed completion(5-1)
Viva(10)			
Able to recall and demonstrate a comprehensive and insightful understanding of the problem domain	Able to recall and demonstrates a comprehensive and insightful understanding of the problem domain(10-8)	Can recall some and demonstrates some understanding of the problem domain(7-5)	Minimal recall and demonstrates a minimal insightful understanding of the problem domain.(4-1)

Course Faculty **Course Coordinator** **Academic Coordinator** **HOD**

Exp No.: 1	Experiments on GSM / GPRS Basic AT Commands, Voice calls / Voice communication, Phone Book, SMS
Date :	

Aim:

To Perform Basic AT Commands, Voice calls / Voice communication, Phone Book, SMS Using GSM module.

Components Required:

SI.No	Item	No's
1	USB-to-Serial (TTL) Converter (FTDI, CP2102, CH340G, etc.)	1
2	GSM Module (SIM800L/SIM900	1
3	JUMPER WIRES	1

PROCEDURE:

STEP 1:

Download & Install PuTTY

STEP 2:

Conenct GSM Module to PC via USB-TTL Converter

STEP 3:

Open PuTTY and Configure Serial Communicatio
Launch PuTTY.Select "Serial" as Connection Type.
Set the correct COM Port: (Check in Device Manager → Ports (COM & LPT)).
Set Baud Rate to 9600 or 115200 (depending on your GSM module).
Click "Open".

STEP 4

Send AT Commands via PuTTY
Once the terminal opens, type AT and press Enter.
Receive the following commands
nginx

CopyEdit
OK
If the response is not received, try pressing Enter again.

Common AT Commands in PuTTY:

Check connection: AT → Response: OK
Check signal strength: AT+CSQ → Response: +CSQ: 20,0
Check network registration: AT+CREG? → Response: +CREG: 0,1 (Registered)

Coding:

```
#include <SoftwareSerial.h>
// Define TX/RX for GSM moduleSoftwareSerial gsm(2, 3); // RX=2, TX=3
void setup() {
Serial.begin(9600);    // Serial monitor communication
gsm.begin(9600);       // GSM module communication

Serial.println("Initializing GSM module...");
delay(1000);

gsm.println("AT"); // Test GSM response
delay(1000);
}
void loop() {
// Check for incoming messages from GSM module
if (gsm.available()) {
Serial.write(gsm.read());
   }

   // Check for input from Serial Monitor
   if (Serial.available()) {
      gsm.write(Serial.read());  // Send command to GSM module
   }
}
```

Result:

Thus the Basic AT Commands, Voice calls / Voice communication, Phone Book, SMS Using GSM module are executed Successfully.

EXP NO:2

Establish and analyze reliable wireless communication in a ZigBee Network

AIM:

To establish and analyze reliable wireless communication between a ZigBee **coordinator** and a **device (end device or router)** for data transmission.

COMPONENTS REQUIRED:

S.No	Item	No's
1.	ZigBee modules (e.g., XBee Series 2 or Series 3)	**1**
2.	Microcontrollers (e.g., Arduino Uno, ESP32, or Raspberry Pi)	**1**
3.	USB-to-Serial converter (if using a PC for the coordinator)	**1**
4.	Sensors (temperature, humidity, soil moisture, etc.)	**1**
	Power supply	

Experiment Steps:

Module Configuration (Using XCTU):

Assign one module as Coordinator:
PAN ID (e.g., 0x1234)
Function set: ZigBee Coordinator API/AT
Channel (e.g., CH = 0x0C)
Assign the second module as End Device (or Router) with same
PAN ID

Function set: ZigBee Router AT/End Device AT
Set Destination Address to the Coordinator's 64-bit address

Hardware Wiring:

Coordinator connected to PC via USB
End device connected to microcontroller and sensor
Use TX/RX for UART communication

Microcontroller Code (End Device):

Read sensor data (e.g., temperature from DHT11)
Send data via serial to ZigBee module
ZigBee module transmits to coordinator

CODING

```
#include oftwareSerial.h>
#include <DHT.h>
#define DHTPIN 2#define DHTTYPE DHT11SoftwareSerial zigbee(10, 11); // RX, TX
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  zigbee.begin(9600);
  dht.begin();
}
void loop() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();
  if (!isnan(temp) && !isnan(hum)) {
    zigbee.print("Temp:");
    zigbee.print(temp);
    zigbee.print("C, Hum:");
    zigbee.print(hum);
    zigbee.println("%");
  }
```

```
  delay(5000);
}
```

RESULT:

Thus the wireless communication between a ZigBee **coordinator** and a **device** for data transmission is established.

EX no: 3	**DESIGN A SIMPLE ALARM SYSTEM**
Date	

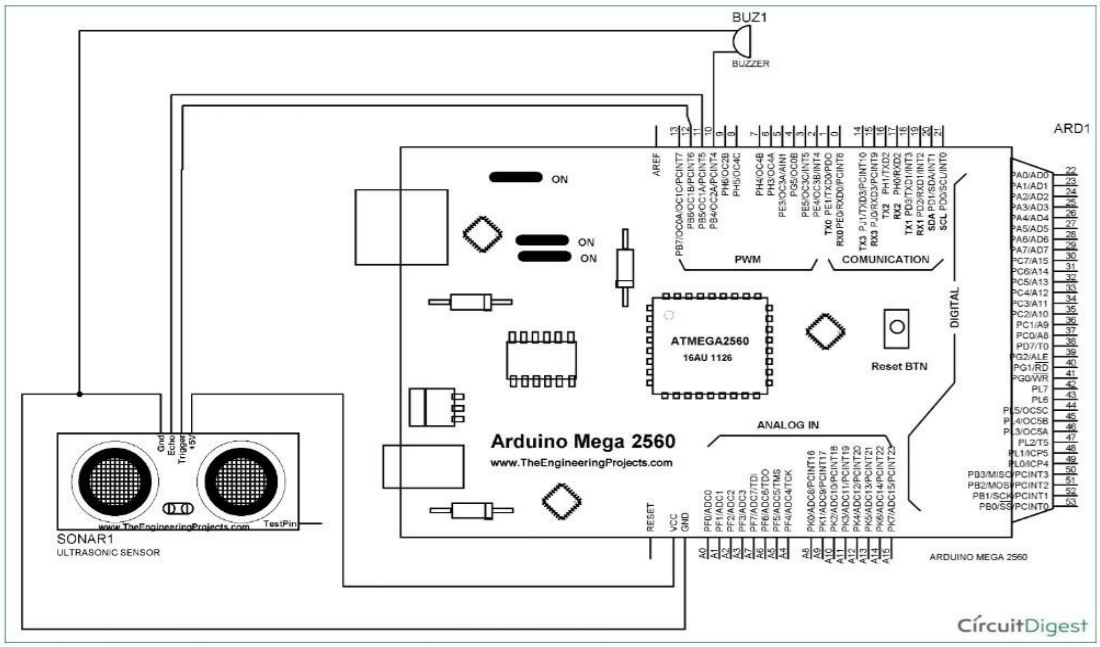
AIM:

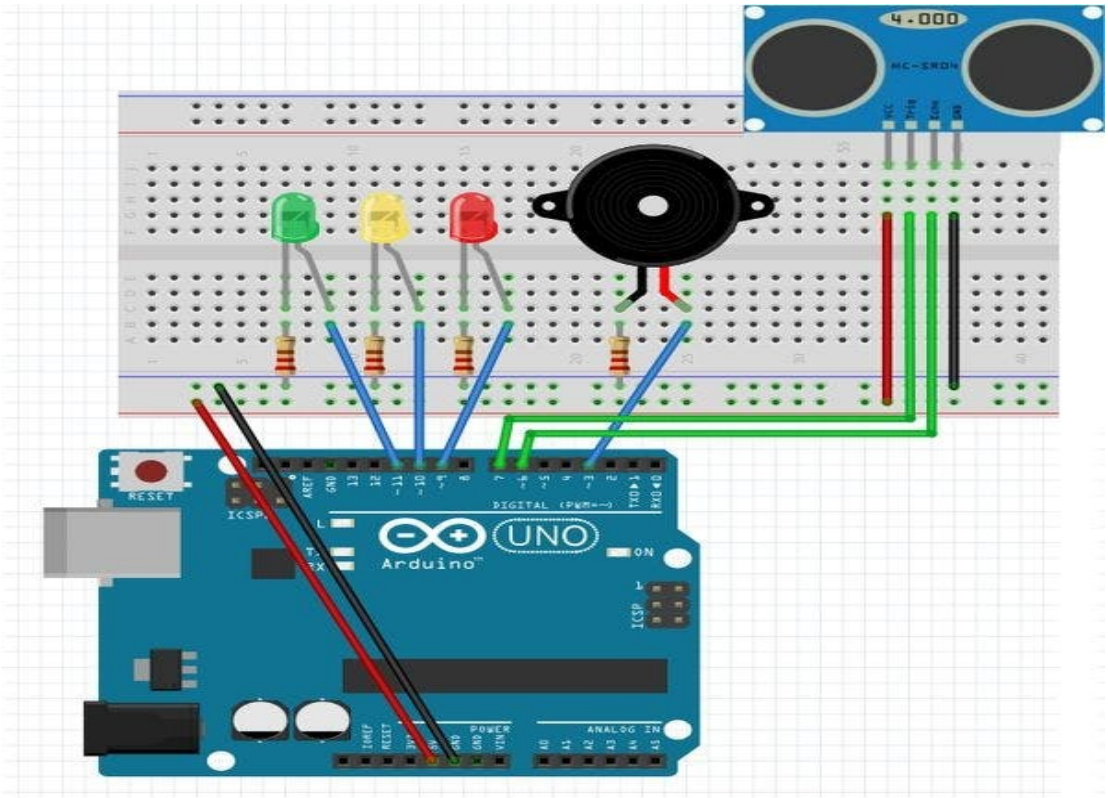
To design a simple alarm system using Ultra sonic sensor HC-SR04 and Arduino.

COMPONENTS REQUIRED:

S.NO	ITEM	No's
1.	Arduino	1
2.	LED	3
3.	Breadboard	1
4.	Resistor	4
5.	Buzzer	1
6.	Ultrasonic Sensor	1
7.	Jump wires	10

BLOCK DIAGRAM:





Procedure:

STEP 1: Setup LED's

Setup the LED lights on the board. Put the Cathode(-) into the negative run of the breadboard and the Anode(+) into a hole next to it.

STEP 2: Buttons

Put the buttons in so you have atleast two pin holes space to add the wires and resistors. Place the 1kohm resistor between one side of the button and negative run on the breadboard.

STEP 3: Wireup LED'S

Wire guide:

LED/Pin

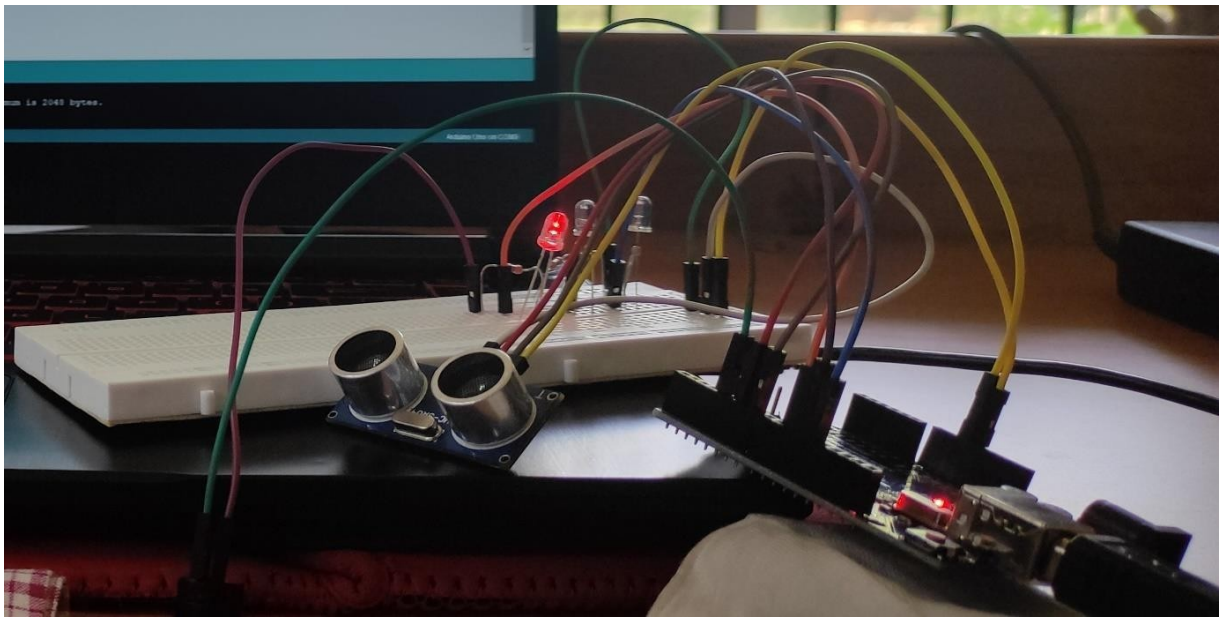
Red – 2

Blue – 1

STEP 4: Upload and run

Connect Arduino to the computer and upload the program. After it has uploaded , it cycles through the lights. When the obstacle distance is reduced, the lights glow accordingly.When the obstacle is too closer, the buzzer sounds.

Implementation:

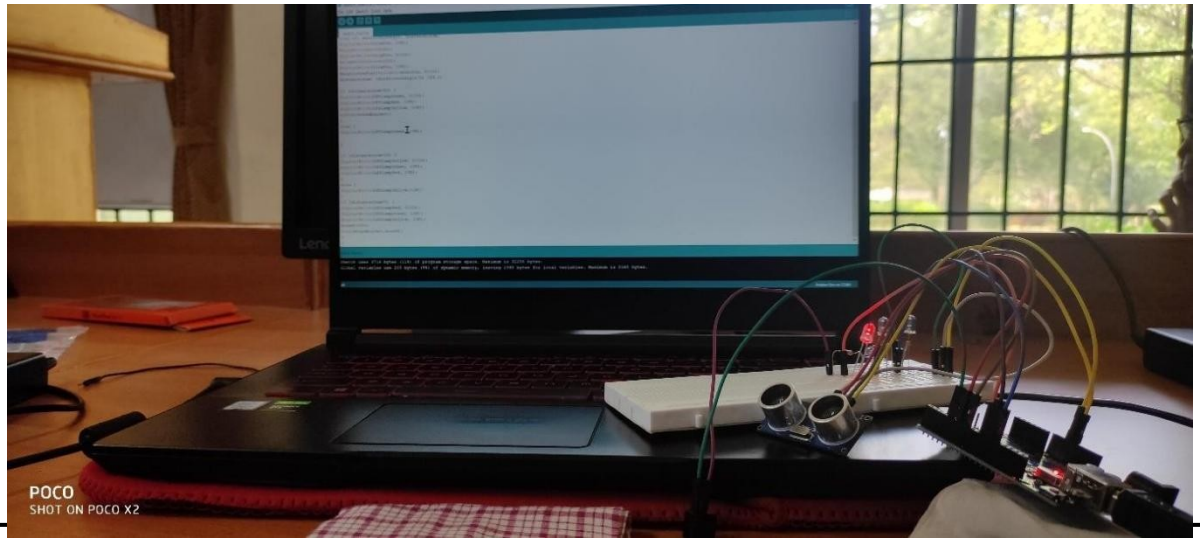


Coding:

```
#define trigPin 7
#define echoPin 6
#define LEDlampRed 10
#define LEDlampYellow 11

#define LEDlampGreen 12
#define soundbuzzer 4
int sound=500;

void setup()
{ Serial.begin
(9600);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(LEDlampRed, OUTPUT);
pinMode(LEDlampYellow, OUTPUT);
pinMode(LEDlampGreen, OUTPUT);
pinMode(soundbuzzer, OUTPUT);
}
void loop() {
long int durationindigit, distanceincm;
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
```

```
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
durationindigit=pulseIn(echoPin, HIGH);
distanceincm= (durationindigit/5) /29.1;



if (distanceincm<50)
{ digitalWrite(LEDlampGreen,
HIGH); digitalWrite(LEDlampRed,
LOW); digitalWrite(LEDlampYellow,
LOW); noTone(soundbuzzer);
}
else {
digitalWrite(LEDlampGreen, LOW);

}
if (distanceincm<20)
{ digitalWrite(LEDlampYellow,
HIGH); digitalWrite(LEDlampGreen,
LOW); digitalWrite(LEDlampRed,
LOW);
}
else {
digitalWrite(LEDlampYellow,LOW);
}
if (distanceincm<5)
{ digitalWrite(LEDlampRed, HIGH);
digitalWrite(LEDlampGreen, LOW);
digitalWrite(LEDlampYellow, LOW);
sound=1000;
tone(soundbuzzer,sound);
}
}
```

Output:



RESULT:

Thus the simple alarm system using Ultrasonic sensors HC-SR04 and Arduino is executed and verified successfully.

Exp no :4	**Web page integration with Raspberry Pi / NODEMCU**
Date:	

AIM:

To create a Web page integration with Raspberry Pi / NODEMCU

COMPONENTS:

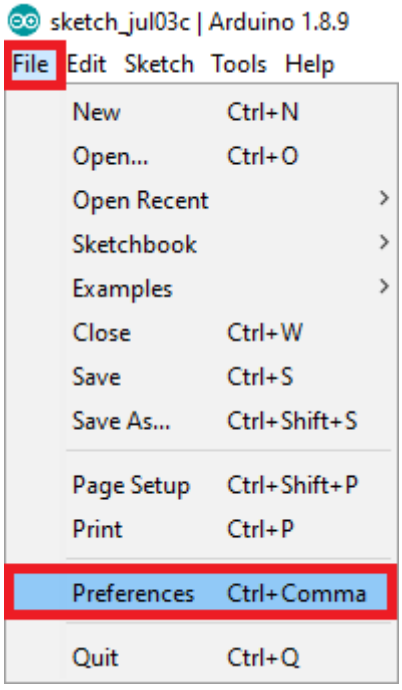
SL.NO	ITEM	NO'S
1	ESP8266 NodeMCU	1
2	Breadboard	1
3	Jumper wires	As required
4	LED	1
5	Arduino IDE	-

IMPLEMENATION:

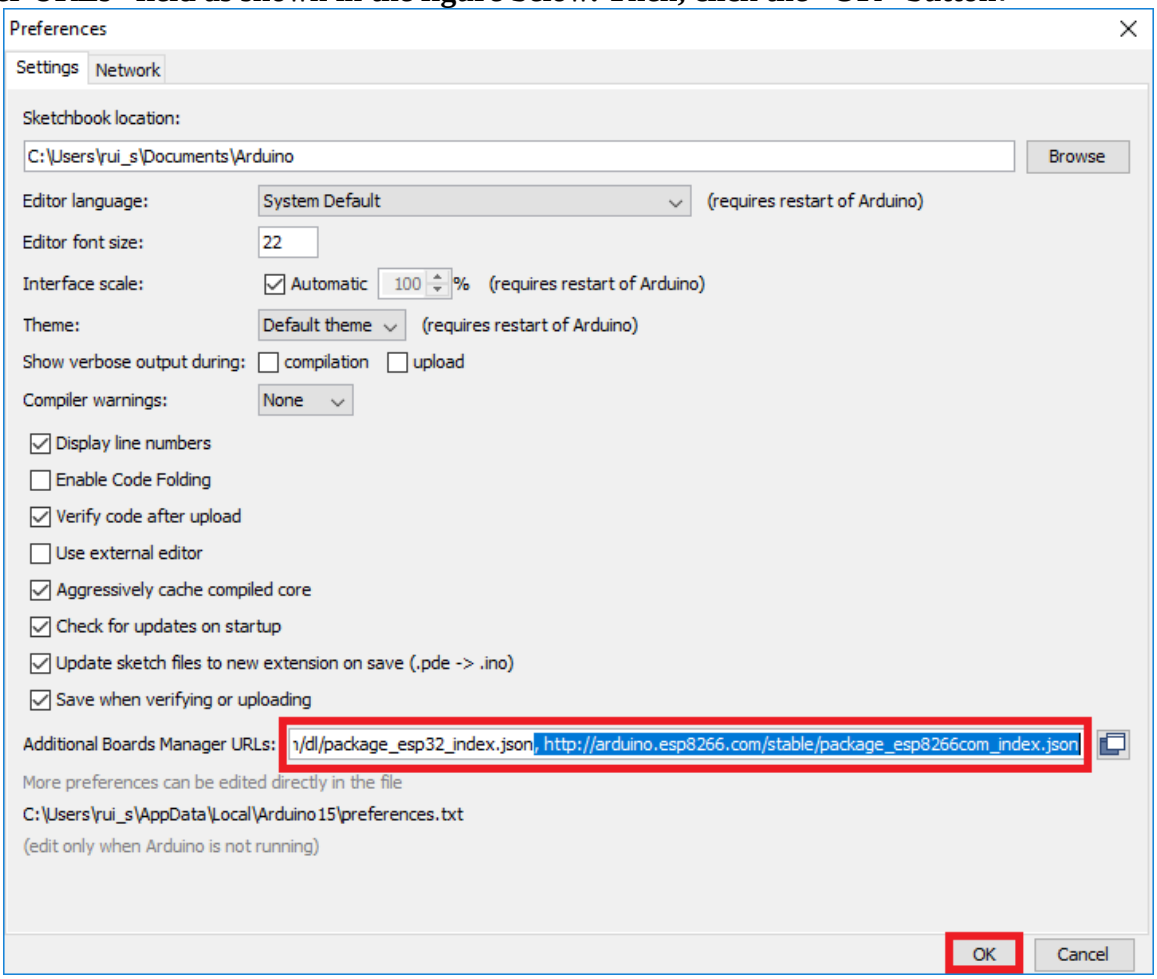
Install ESP8266 Add-on in Arduino IDE

To install the ESP8266 board in your Arduino IDE, follow these next instructions:

In your Arduino IDE, go to File> Preferences

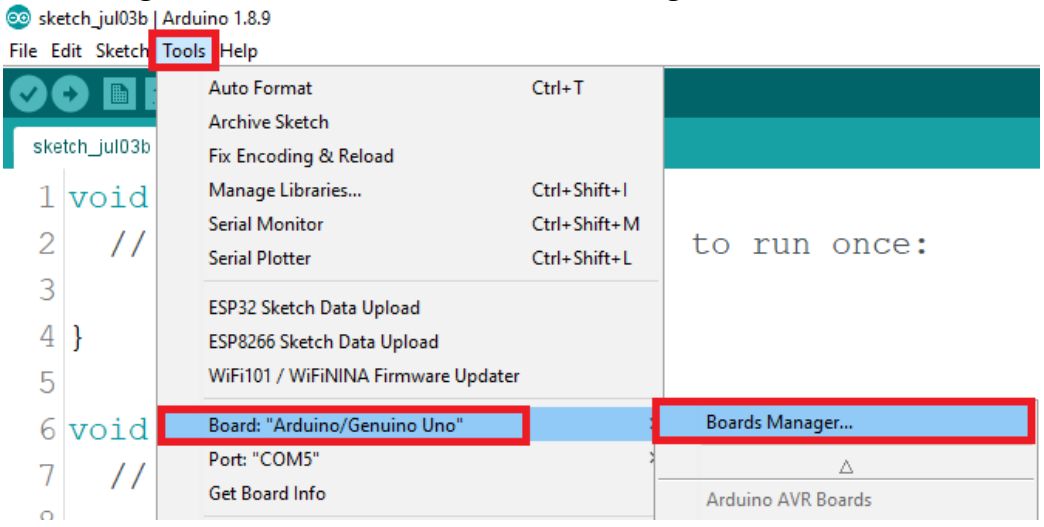


1. **Enter http://Arduino.esp8266.com/stable/package_esp8266com_index.json into the "Additional Boards Manager URLs" field as shown in the figure below. Then, click the "OK" button**:

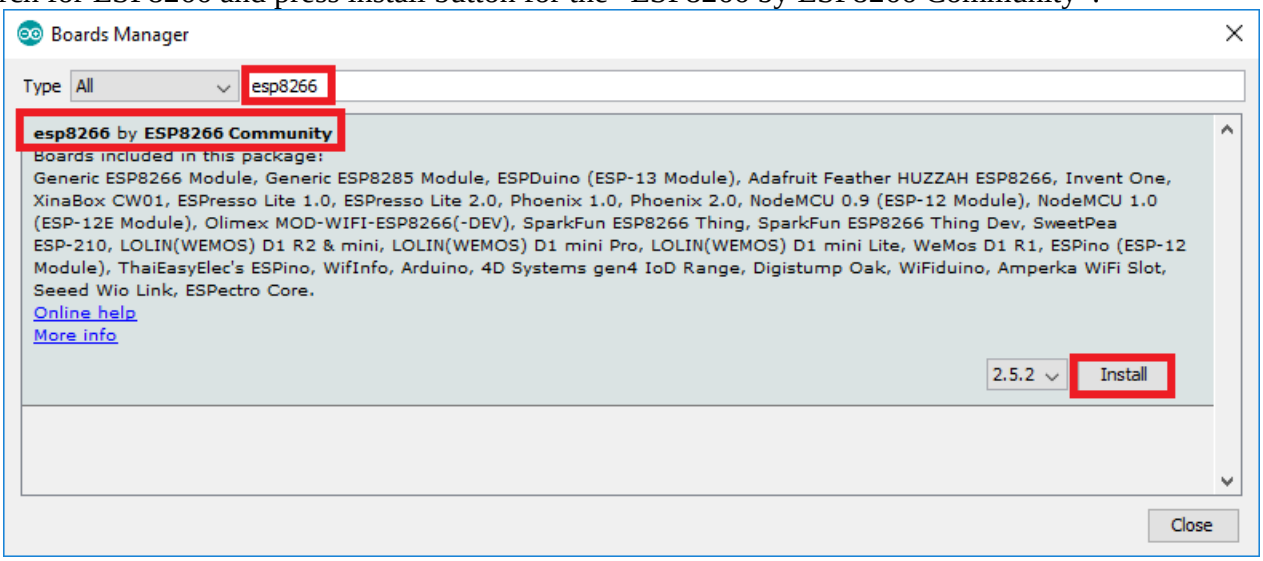


Note: if you already have the ESP32 boards URL, you can separate the URLs with a comma as follows: https://dl.espressif.com/dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json

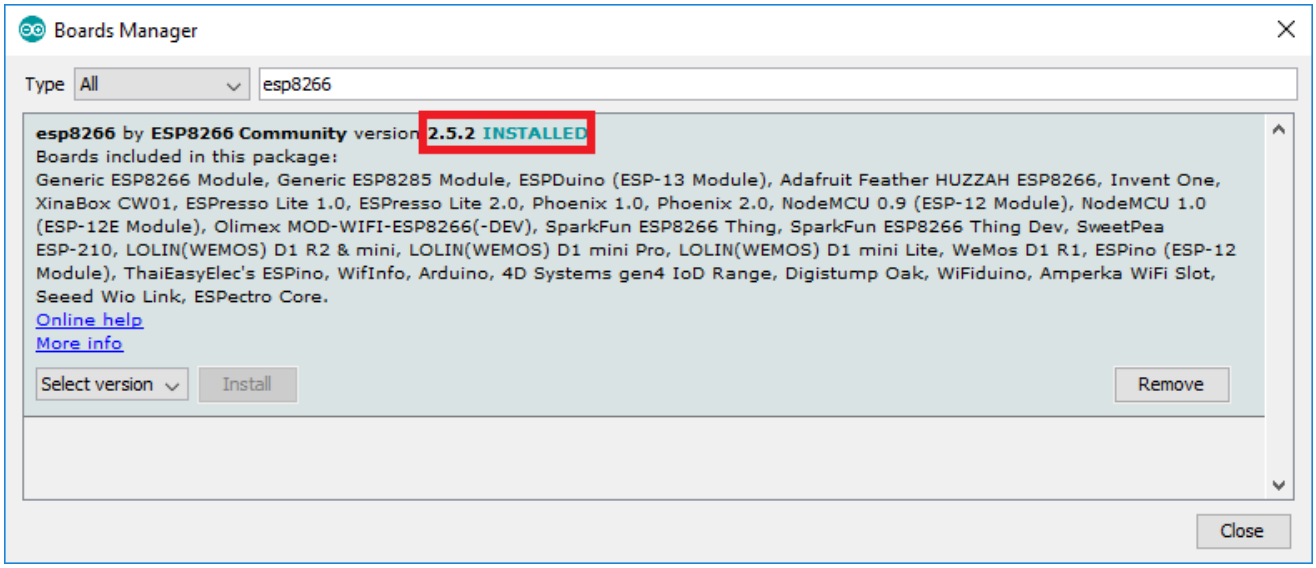
2. Open the Boards Manager. Go to Tools > Board > Boards Manager…

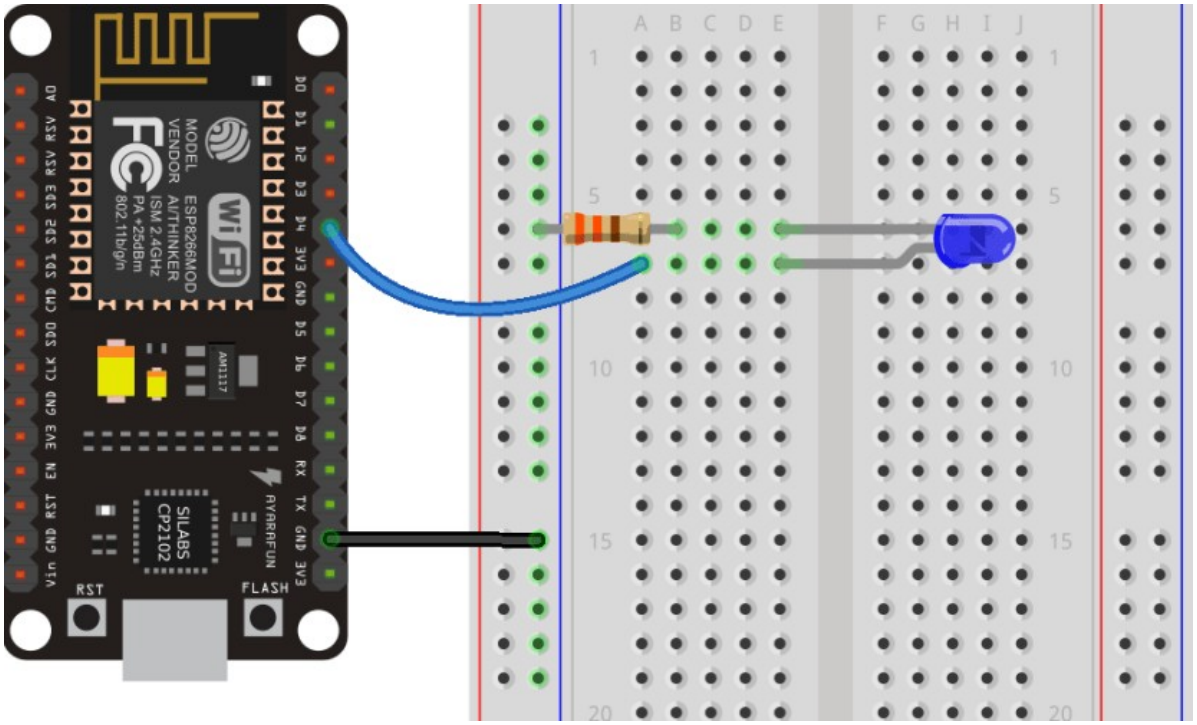


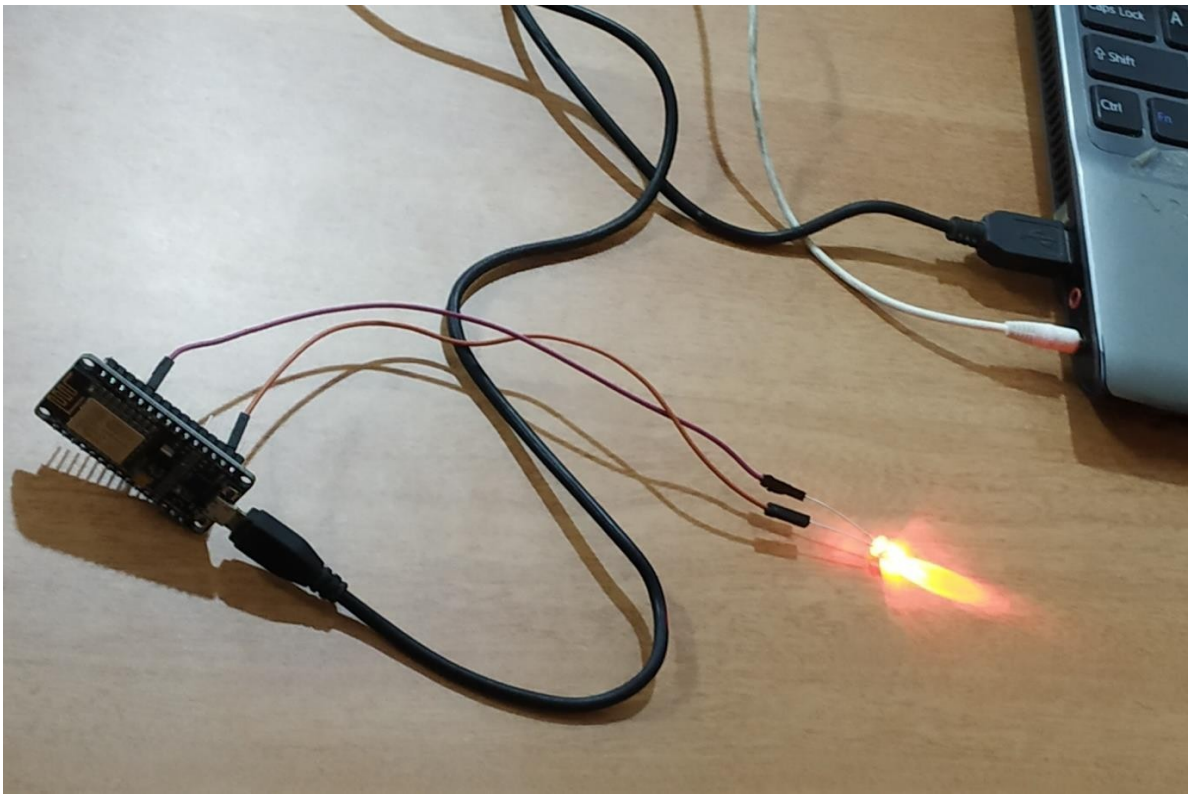
3. Search for ESP8266 and press install button for the “ESP8266 by ESP8266 Community“:



4. That’s it. It should be installed after a few seconds.




A B C D E
F G H I J
1
5
10
15
20
MODEL ESP8266MOD
VENDOR AITHINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n
Wi Fi
SILABS CP2102
RST
FLASH



Step1: Installation

Install ESP8266 Add-on in Arduino IDE.

Step 2: Setup LED

Connect the LED directly to the wire and connect the wire to pin 2 of NodeMCU.

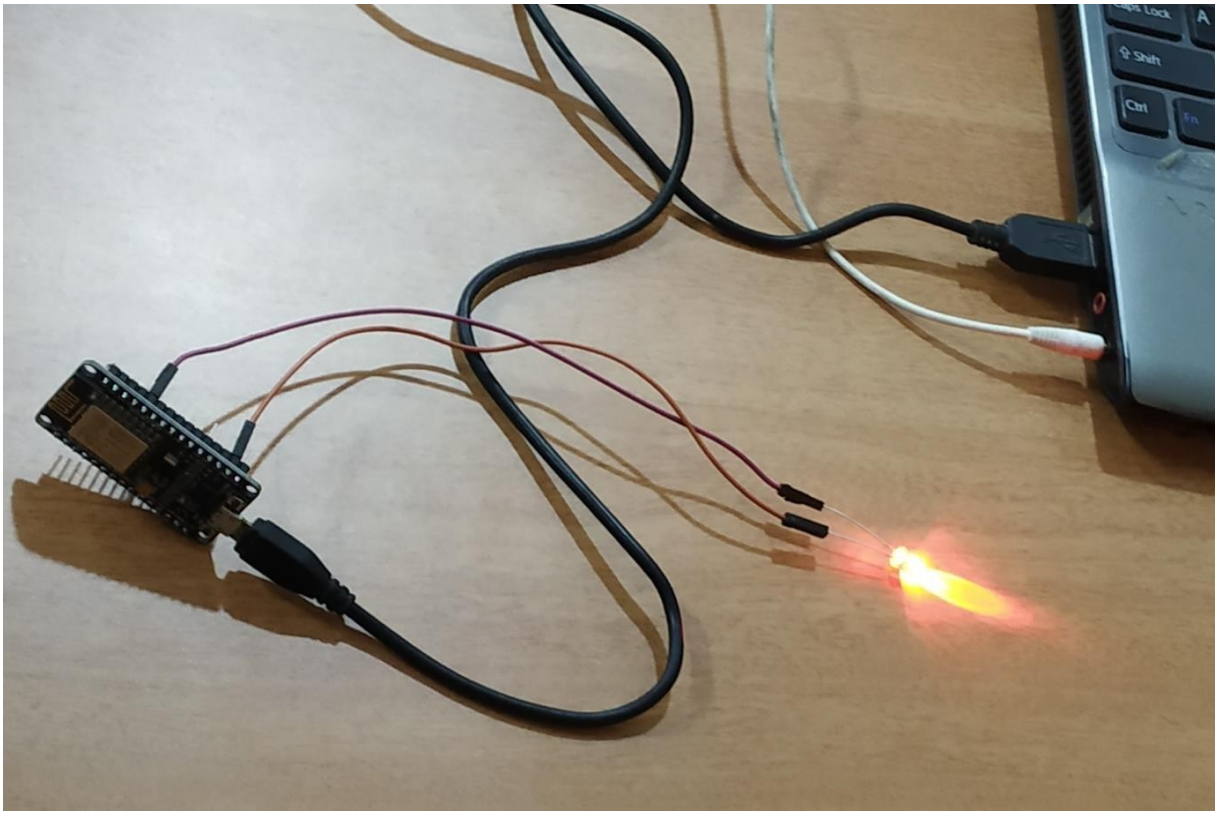
Step 3: Program

```
int pin =2;
voidsetup(){
// initialize GPIO 2 as an output.
pinMode(pin, OUTPUT);
}
// the loop function runs over and over again forever
voidloop(){
digitalWrite(pin, HIGH);// turn the LED on (HIGH is the voltage level)
delay(1000);// wait for a second
digitalWrite(pin, LOW);// turn the LED off by making the voltage LOW
delay(1000);// wait for a second
}
```

Step 4: Upload and run

Hold-down the BOOT/FLASH button in your ESP8266 development board**.** Then press the Upload button in the Arduino IDE to upload your sketch**.** When you see the Connecting…. message in your Arduino IDE, release the finger from the BOOT/FLASH button**.** After that, you should see the Done uploading message and your ESP8266 should have the new sketch running. Press the ENABLE/RESET button to restart the ESP8266 and run the new uploaded sketch. After it has uploaded the light will turn on.

Output:



RESULT:

Thus the Web page is integrated with Raspberry Pi / NODEMCU

Exp no :5 Create your own smart light using Raspberry Pi or Arduino Uno.

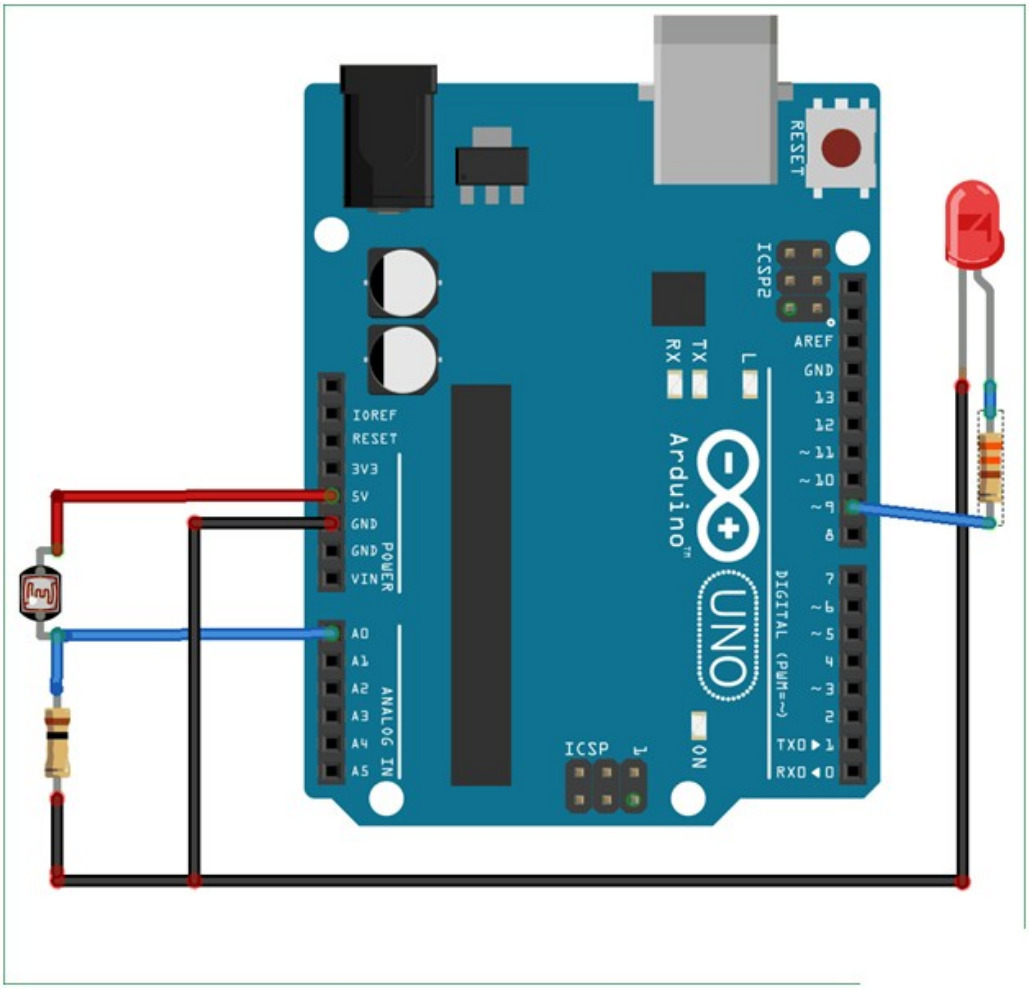
AIM:

To create your own smart light using Raspberry Pi or Arduino Uno.

COMPONENTS REQUIRED:

S.NO	ITEM	NO's
1.	Arduino UNO	1
2.	LDR(Light Dependent Resistor)	1
3.	Resistor	2
4.	LED	1
5.	Relay module-5v	1
6.	Bulb/CFL	1
7.	Connecting wires	7
8.	Breadboard	1

CIRCUIT DIAGRAM:



PROCEDURE:

STEP 1: Connect the 3.3v output of the Arduino to the positive rail of the breadboard
STEP 2: Connect the ground to the negative rail of the breadboard

STEP 3: Place the LDR on the breadboard

STEP 4: Attach the 10K resistor to one of the legs of the LDR

STEP 5: Connect the A0 pin of the Arduino to the same column where the LDR and resistor is connected (Since the LDR gives out an analog voltage, it is connected to the analog input pin on the Arduino. The Arduino, with its built-in ADC (Analog to Digital Converter), then converts the analog voltage from 0-5V into a digital value in the range of 0-1023). connect the other end of the 10K resistor to the negative rail

STEP 6: And the the second (free) leg of the LDR to the positive rail

STEP 7: Place the LED on the breadboard

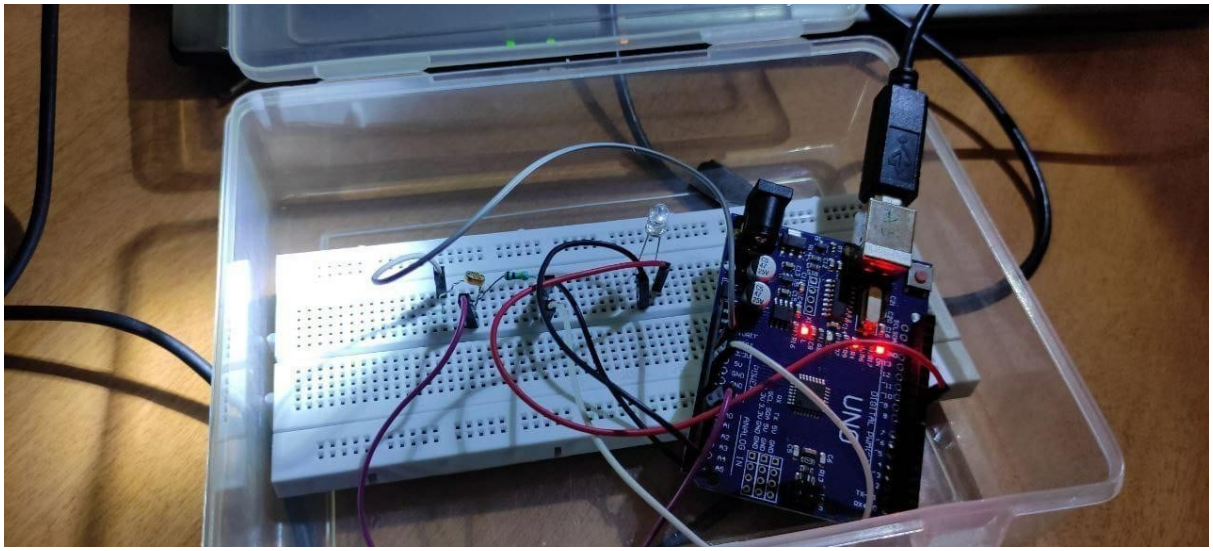
STEP 8: Connect the 220ohm resistor to the long leg (+ve) of the LED

STEP 9: Then we will connect the other leg of the resistor to pin number 13 (digital pin) of the Arduino

STEP 10: The shorter leg of the LED to the negative rail of the breadboard

Basic circuits like this can be done without an Arduino aswell .We want our circuit to do something in the real world other than just displaying the values on the computer screen we will be attaching a LED to the circuit. The LED will turn on when its dark and will go off when its bright.

IMPLEMENTATION:

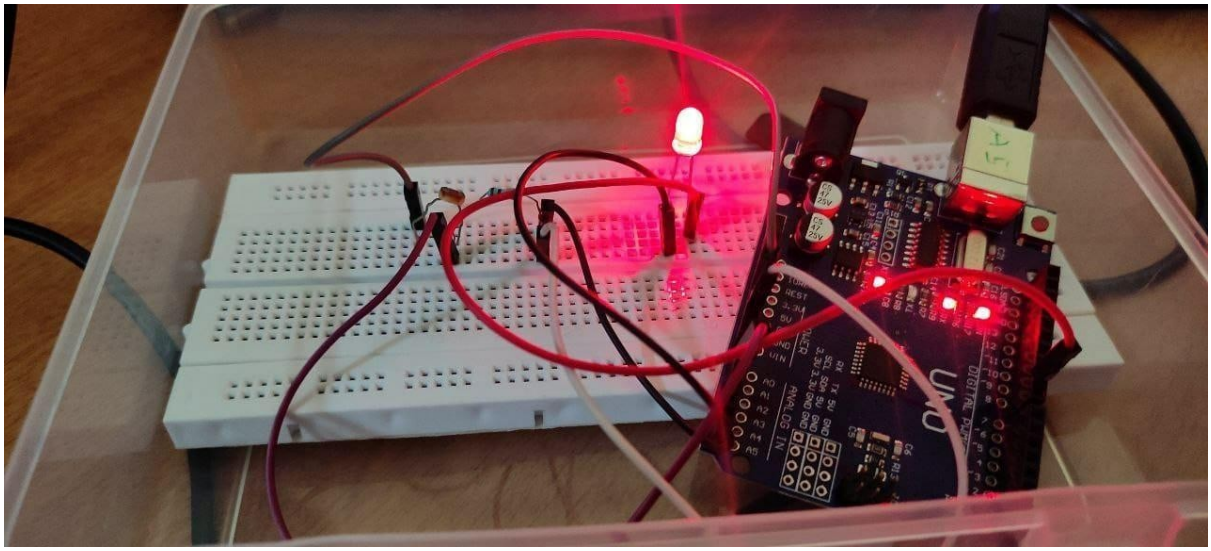


CODING:

```
#define relay 10
int LED = 9; int
LDR = A0; void
setup()
{
Serial.begin(9600); pinMode(LED,
OUTPUT); pinMode(relay,
OUTPUT); pinMode(LDR,
INPUT);
} void loop()
{
int LDRValue = analogRead(LDR);
Serial.print("sensor = ");
Serial.print(LDRValue); if
```

```
(LDRValue <=700)
{ digitalWrite(LED,
HIGH); digitalWrite(relay,
HIGH);
Serial.println("It's Dark Outside; Lights status: ON");
} else
{ digitalWrite(LED,
LOW); digitalWrite(relay,
LOW);
Serial.println("It's Bright Outside; Lights status: OFF");
}
}
```

OUTPUT:



RESULT:

Thus the smart light using Raspberry Pi / Arduino Uno is executed and

EX NO: 6 **Date**	Control And Monitor The Temperature of the Elements using Temperature sensor.

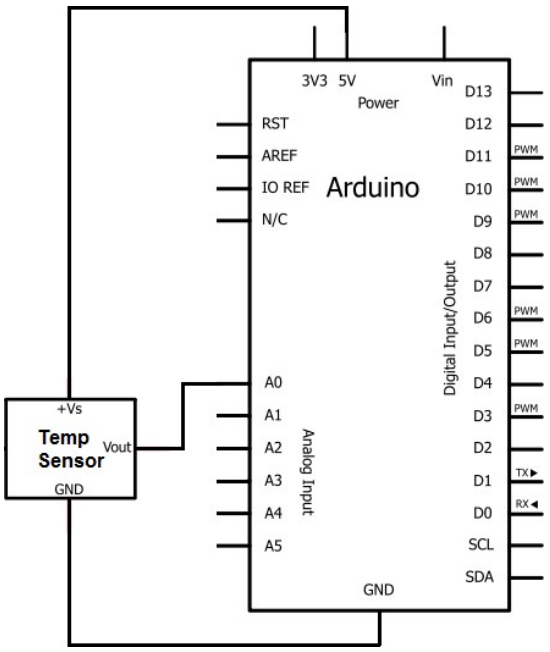
Aim

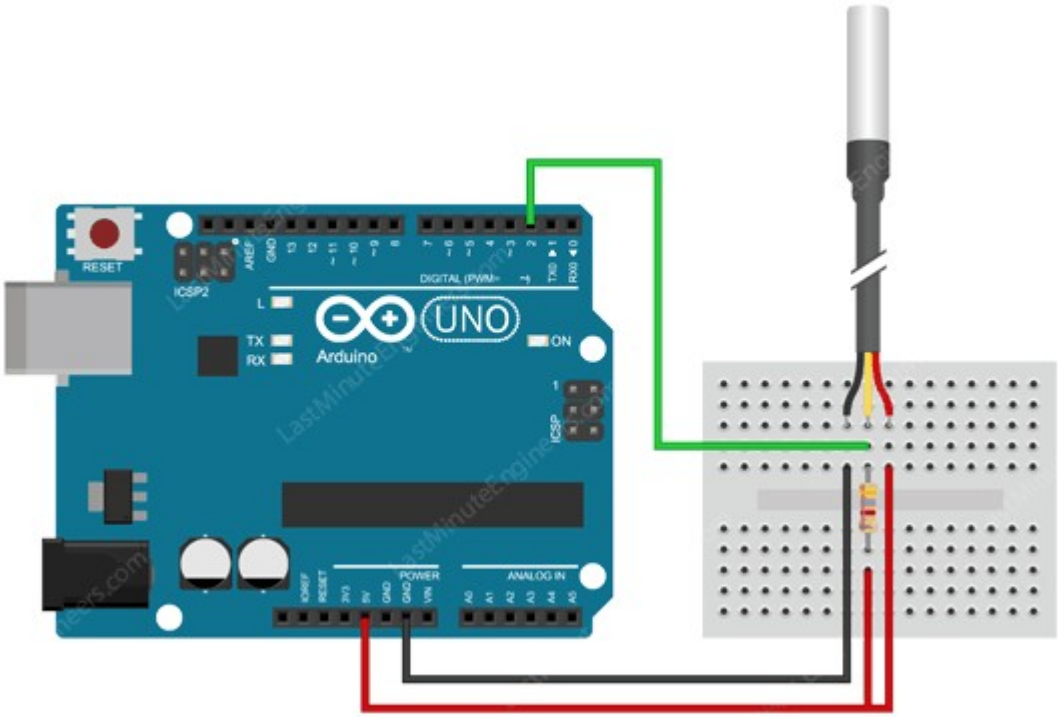
To Control And Monitor The Temperature of the Elements using Temperature sensor.

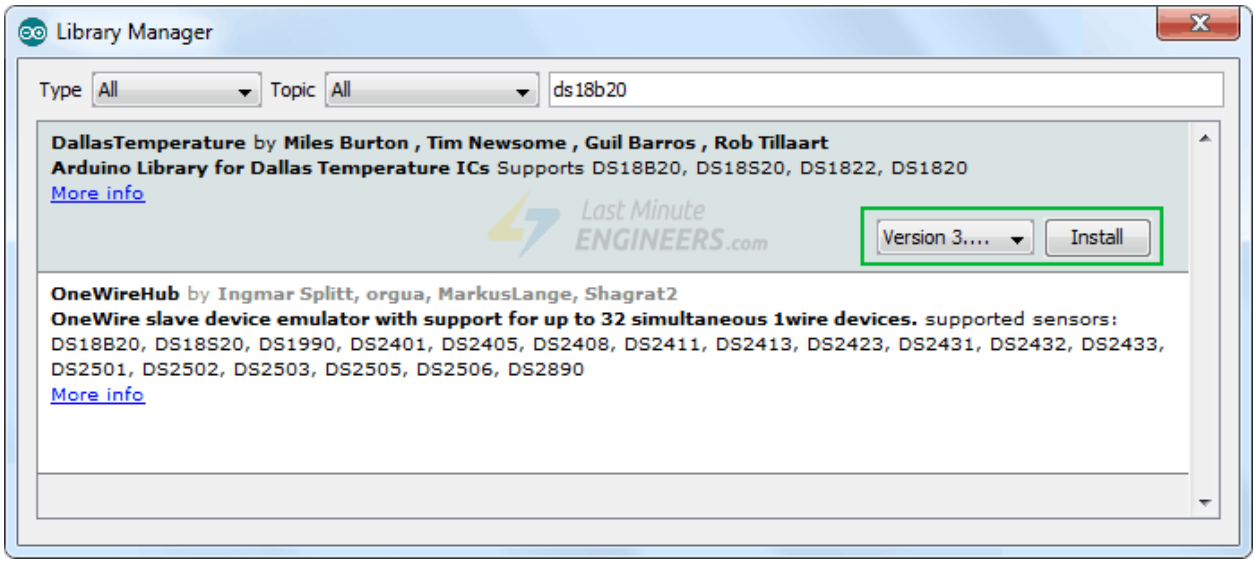
Components Required:

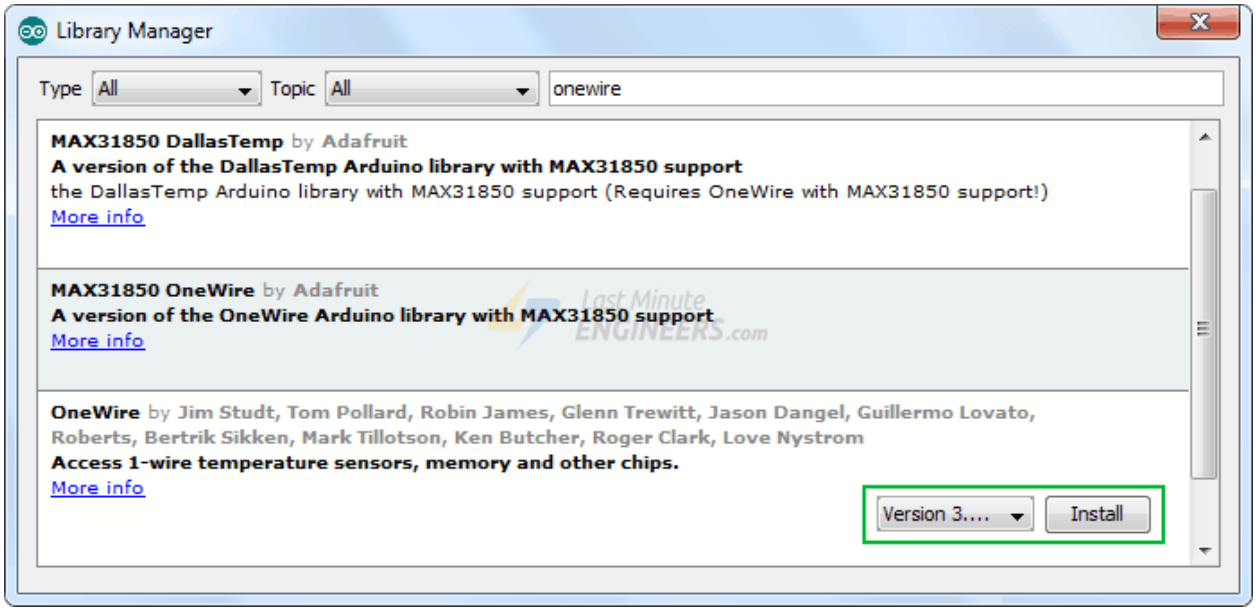
SI.No	Item	No's
1	Breadboard	1
2	Arduino	1
3	Resistor	1
4	Jumper Wires	4
5	Temperature sensor	1

Block Diagram:




RESET
ICSP2
DIGITAL (PWM~)
UNO
Arduino
TX
RX
ON
ICSP
POWER
ANALOG IN


Library Manager
Type All
Topic All
ds18b20
DallasTemperature by Miles Burton , Tim Newsome , Guil Barros , Rob Tillaart
Arduino Library for Dallas Temperature ICs Supports DS18B20, DS18S20, DS1822, DS1820
More info
Last Minute ENGINEERS.com
Version 3....
Install
OneWireHub by Ingmar Splitt, orgua, MarkusLange, Shagrat2
OneWire slave device emulator with support for up to 32 simultaneous 1wire devices. supported sensors:
DS18B20, DS18S20, DS1990, DS2401, DS2405, DS2408, DS2411, DS2413, DS2423, DS2431, DS2432, DS2433,
DS2501, DS2502, DS2503, DS2505, DS2506, DS2890
More info



PROCEDURE:

STEP 1: Setup sensor

Set up the temperature sensor on the breadboard . It has yellow wire(Data line), black wire for ground and Red for Vcc(Power).

STEP 2: Wire up Sensor

Wire guide:

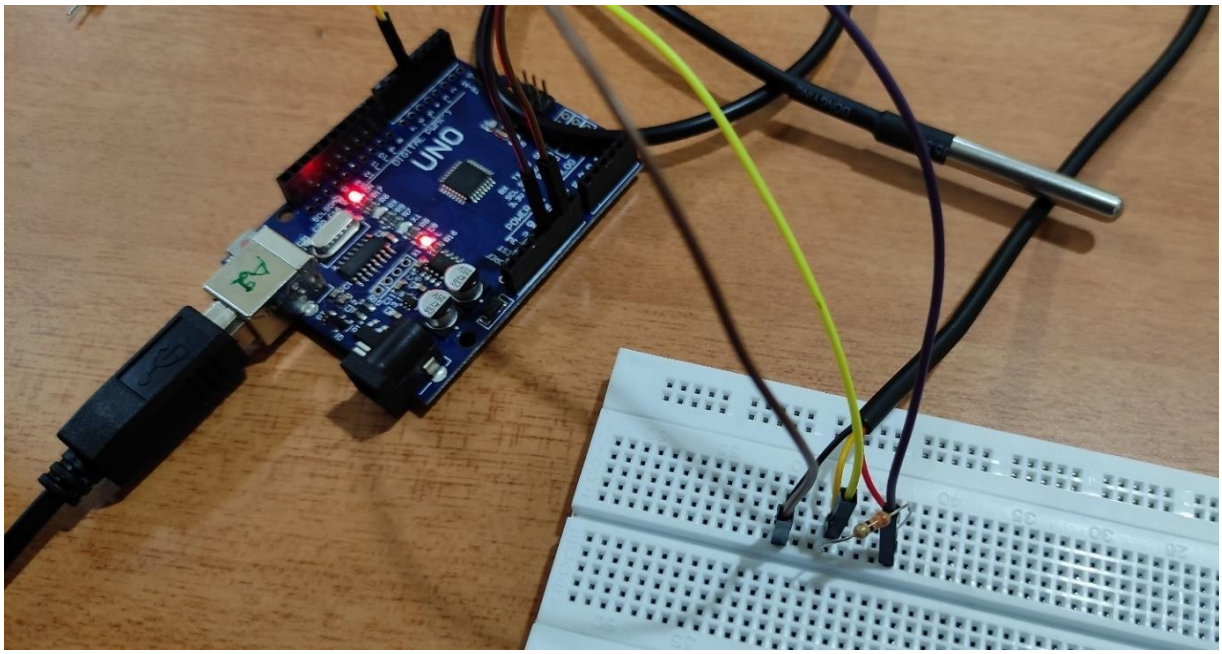
Red Wire – Vcc

Black wire – Gnd

Yellow – Data Input

Connect these temperature sensor to the respective pins in the Arduino.

IMPLEMENTATION:

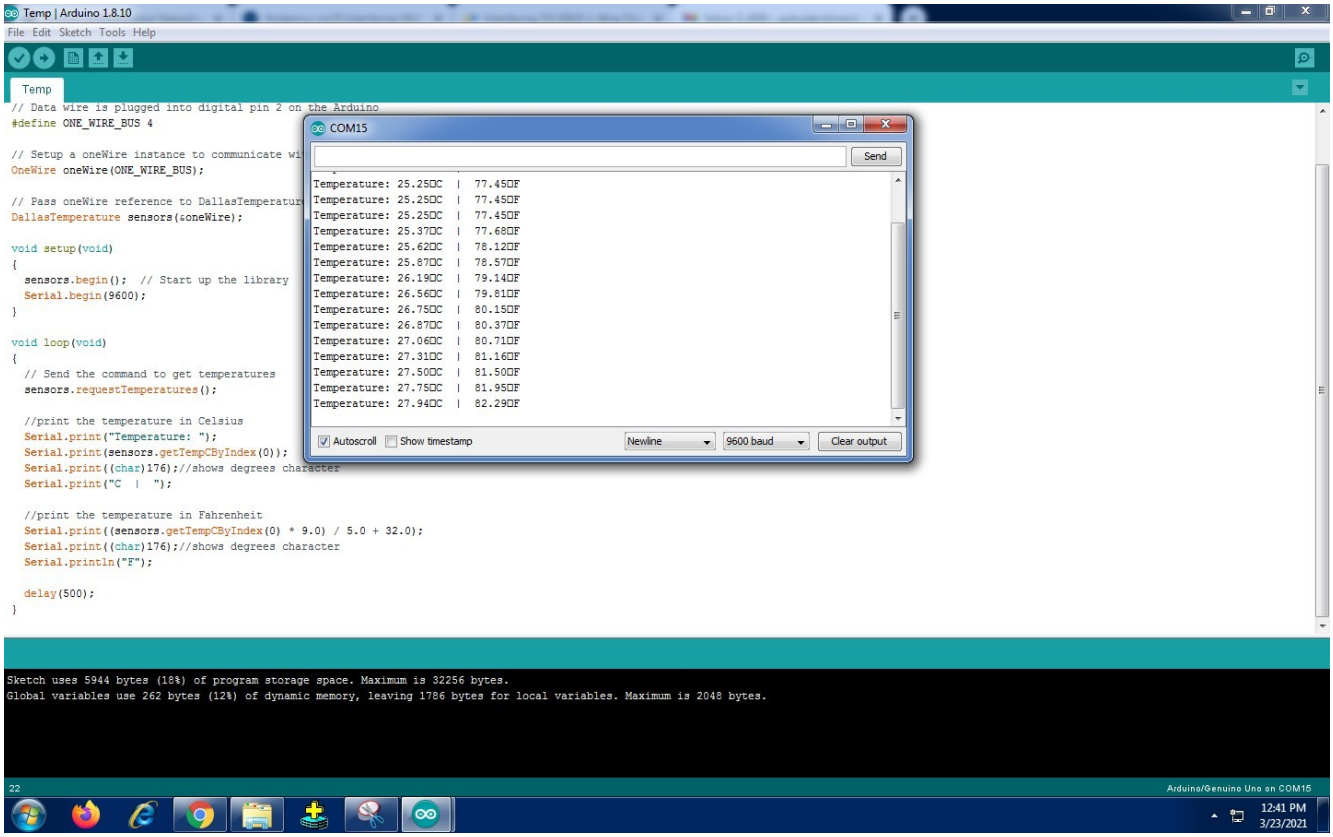


Coding:

```
#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is plugged into digital pin 2 on the Arduino
#define ONE_WIRE_BUS 4

// Setup a oneWire instance to communicate with any OneWire device
OneWire oneWire(ONE_WIRE_BUS);

// Pass oneWire reference to DallasTemperature library
DallasTemperature sensors(&oneWire);

void setup(void)
```

```
{
  sensors.begin(); // Start up the library
  Serial.begin(9600);
}

void loop(void)
{
  // Send the command to get temperatures
  sensors.requestTemperatures();

  //print the temperature in Celsius
  Serial.print("Temperature: ");
  Serial.print(sensors.getTempCByIndex(0));
  Serial.print((char)176);//shows degrees character
  Serial.print("C |  ");

  //print the temperature in Fahrenheit
  Serial.print((sensors.getTempCByIndex(0) * 9.0) / 5.0 + 32.0);
  Serial.print((char)176);//shows degrees character
  Serial.println("F");

  delay(500);
}
```

Output:



Result:

The temperature of the element is calibrated successfully using the temperature sensor.

.

Exp no :7 USE SMTP FOR MONITORING POLLUTION LEVELS USING RASPBERRY PI AND PYTHON

Date

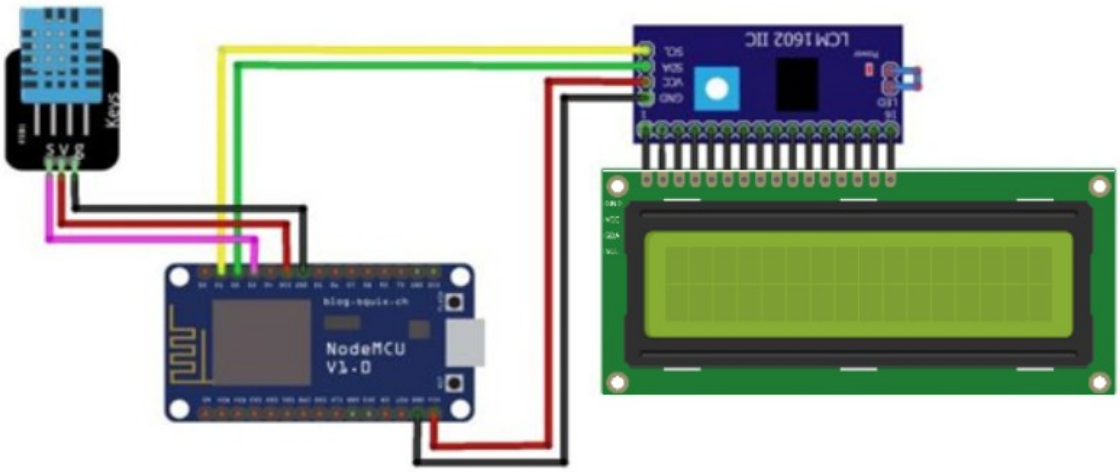
AIM:

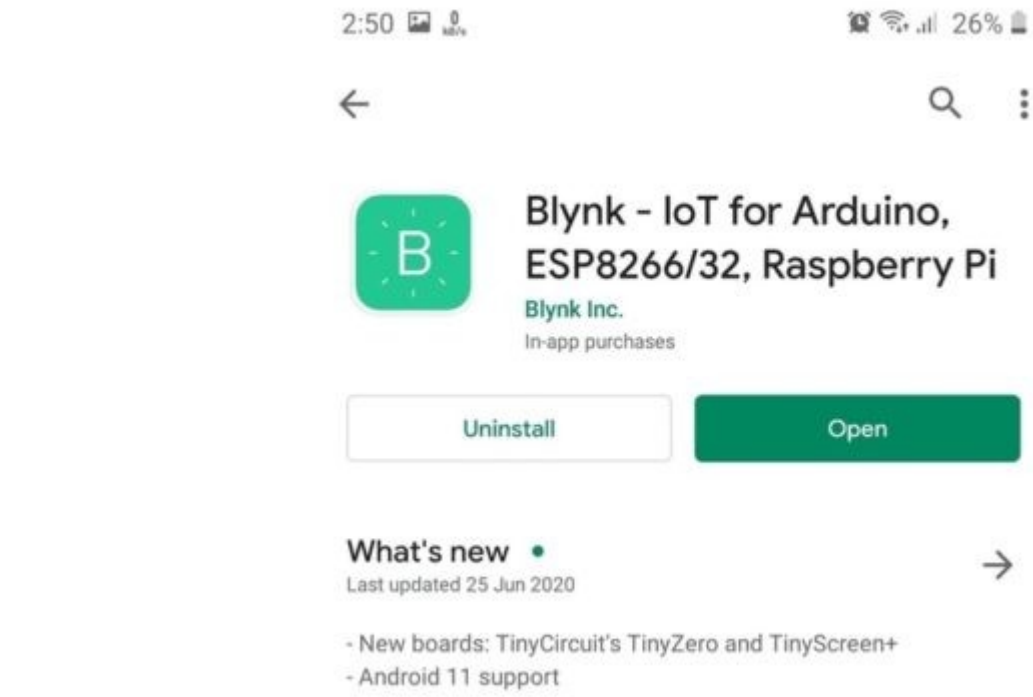
To design our own smart light using Raspberry pi / Arduino uno.

COMPONENTS:

SL.NO	ITEM	NO'S
1	NodeMCU	1
2	LCD Display	1
3	DHT sensor	1
4	Hook-up-wires	As required
5	I2C module	1

IMPLEMENATION:

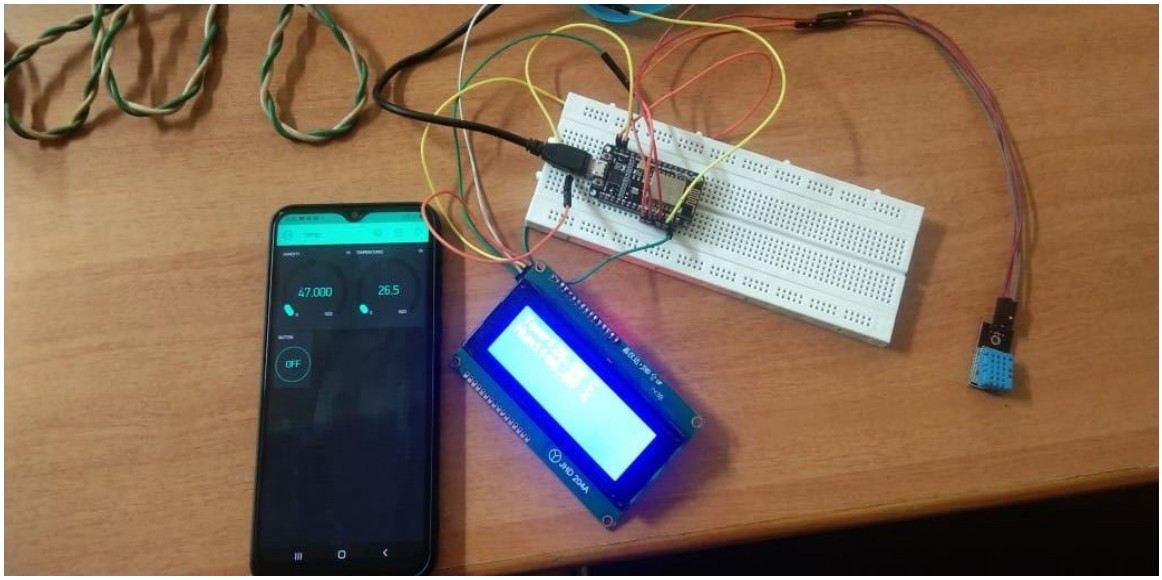



2:50
26%
Blynk - IoT for Arduino, ESP8266/32, Raspberry Pi
Blynk Inc.
In-app purchases
Uninstall
Open
What's new
Last updated 25 Jun 2020
- New boards: TinyCircuit's TinyZero and TinyScreen+
- Android 11 support


Project Settings
Home Automation
OFF
ON
Generate Link
1000
How It Works
Add Shortcut
Home Automation
NodeMCU (Wi-Fi)
Email all
Copy all
DARK
LIGHT


Home Automation
D6
D2
D1
D5


Home Automation
ON
OFF
OFF
ON



Step 1:

Connect the DHT sensor with the NODEMCU. Connect the VCC and GND of NODEMCU with the I2C Module.

Step 2:

Connect the NODEMCU to the computer and upload the program. After it has uploaded it will display the temperature and humidity in the mobile application

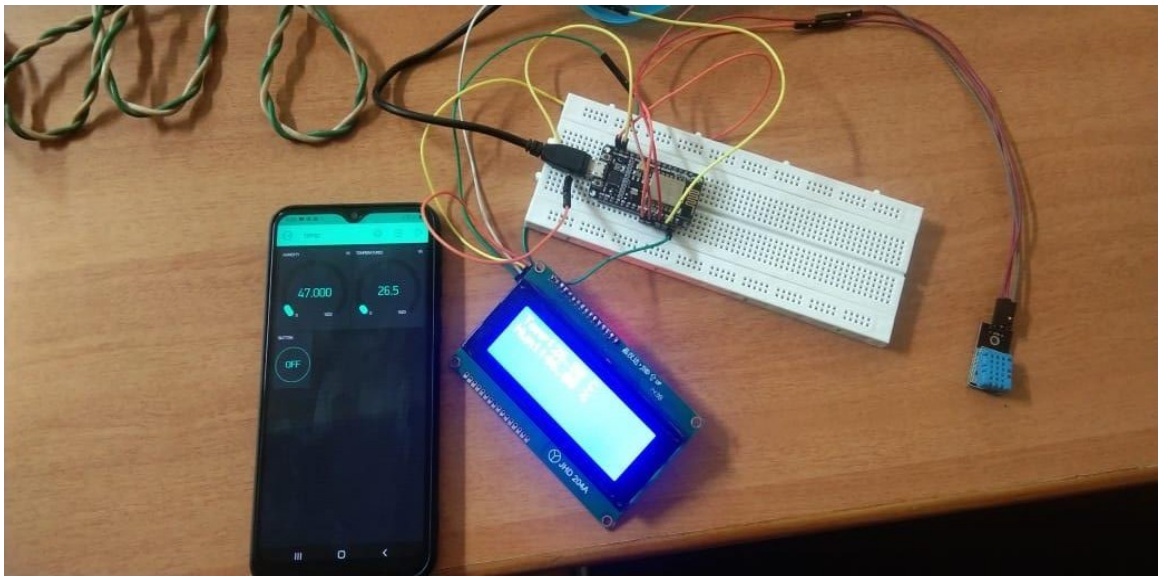
PROGRAM:

```
define BLYNK_PRINT Serial

#include <SPI.h>

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

#include <DHT.h>

#include <Wire.h>

#include <LiquidCrystal_I2C.h>

char auth[] = "RoU-I-dHLk6-JfANUH6_1rYUA9L3NF-v"; //Enter the Auth code which was send by Blink

char ssid[] = "270"; //Enter your WIFI Name

char pass[] = "nuqa4450"; //Enter your WIFI Password

#define DHTPIN D3          // Digital pin 3

#define DHTTYPE DHT11

//#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()

{

  Serial.begin(9600);

  Blynk.begin(auth, ssid, pass);

  lcd.init();

   lcd.backlight();

  dht.begin();

  lcd.begin(16,2);

}

void loop()

{

  lcd.backlight();

  lcd.setCursor(0, 0);
```

```
  lcd.print("Temp:");
  lcd.setCursor(11, 0);
  lcd.print("C");
  lcd.setCursor(0, 1);
  lcd.print("Humi:");
  lcd.setCursor(11, 1);
  lcd.print("%");
  Sensor();
  Blynk.run(); // Initiates Blynk
}
void Sensor()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
   if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Blynk.virtualWrite(V5, h); //V5 is for Humidity
  Blynk.virtualWrite(V6, t); //V6 is for Temperature
  lcd.setCursor(5, 0);
  lcd.print(t);
  lcd.setCursor(5, 1);
  lcd.print(h);
  delay(1000);
}
```

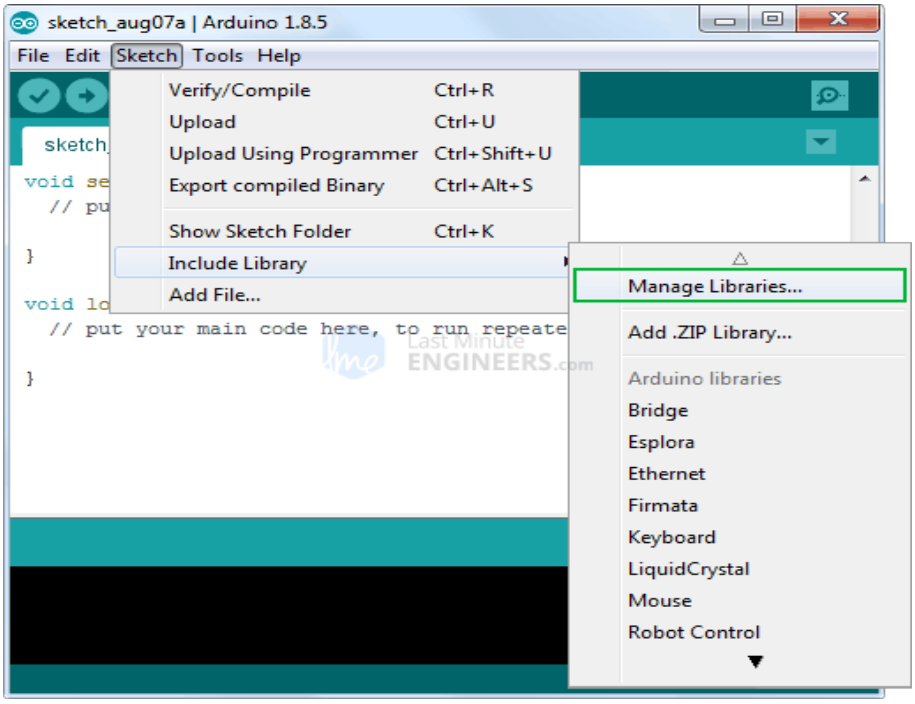
.

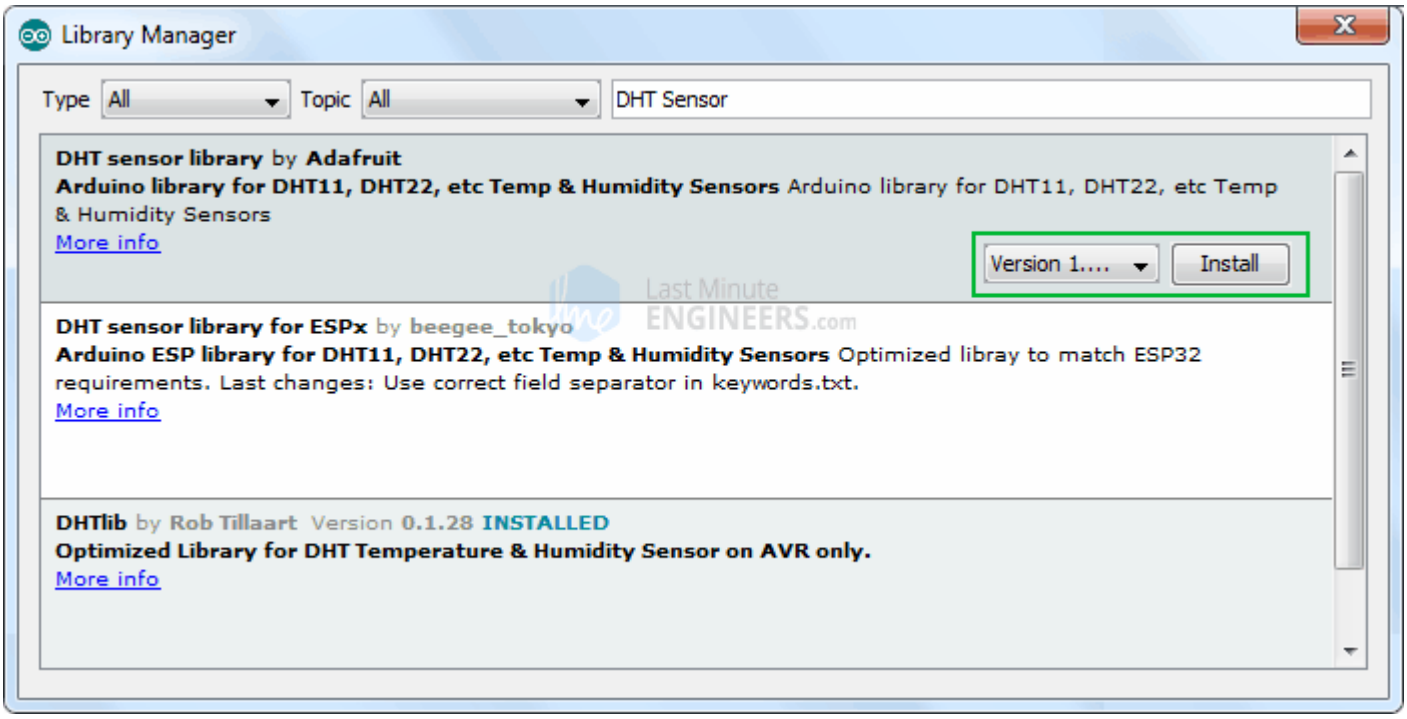
Output:



Installing DHT Sensor Library

To install the library navigate to the **Sketch > Include Library > Manage Libraries…** Wait for Library Manager to download libraries index and update list of installed libraries.





RESULT:

Thus, designing our own smart light using Raspberry pi / Arduino uno was completed successfully.

EXP NO:8

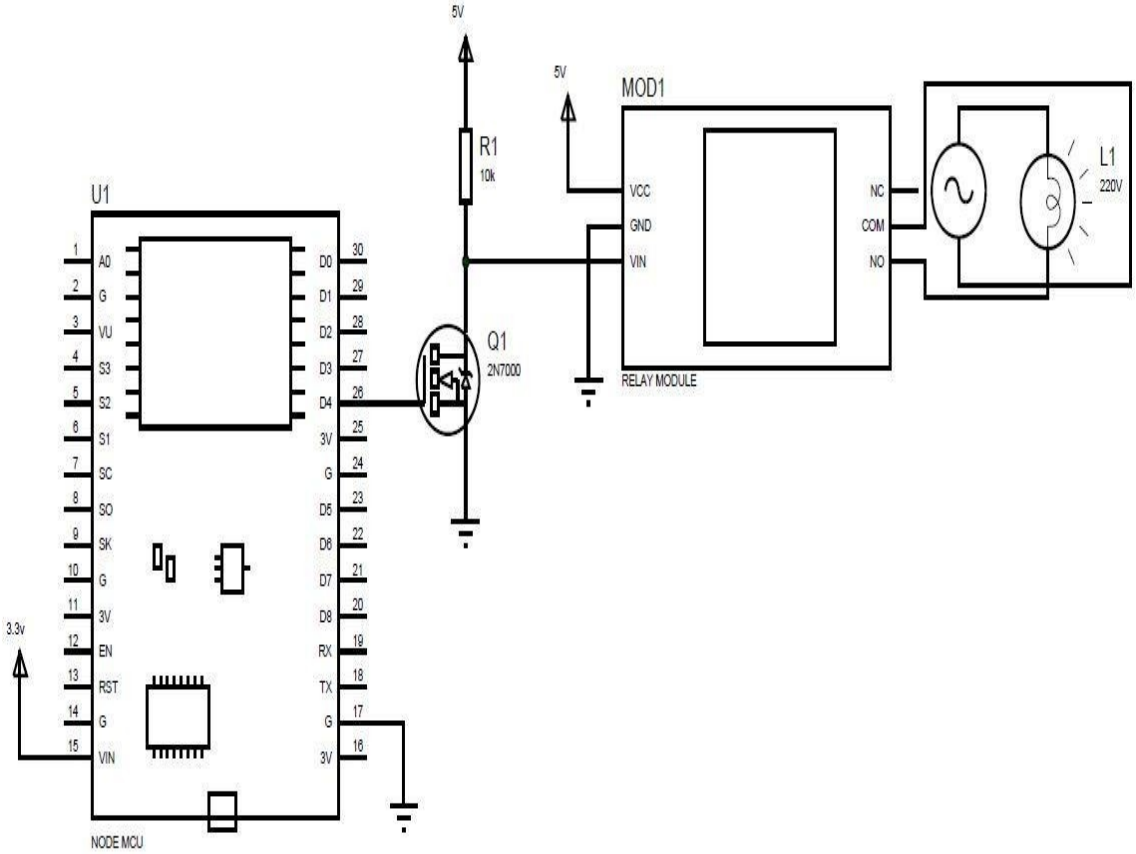
AIM: CONTROL ANY ELECTRICAL APPLIANCE VIA WEBPAGE USING NODEMCU

Date: To design and control a bulb via webpage using NODEMCU ESP8266 .

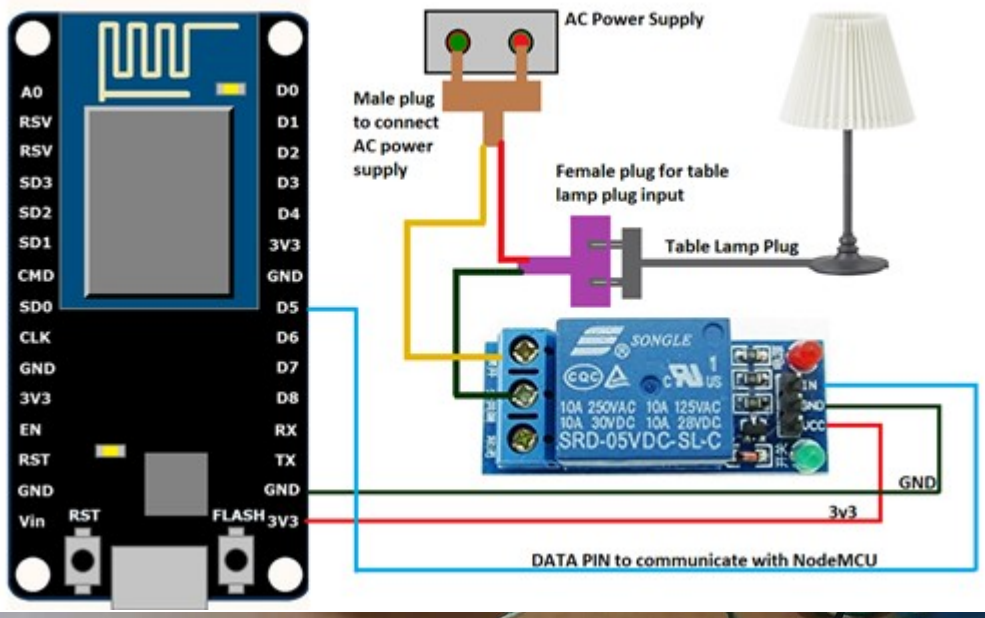
COMPONENTS REQUIRED:

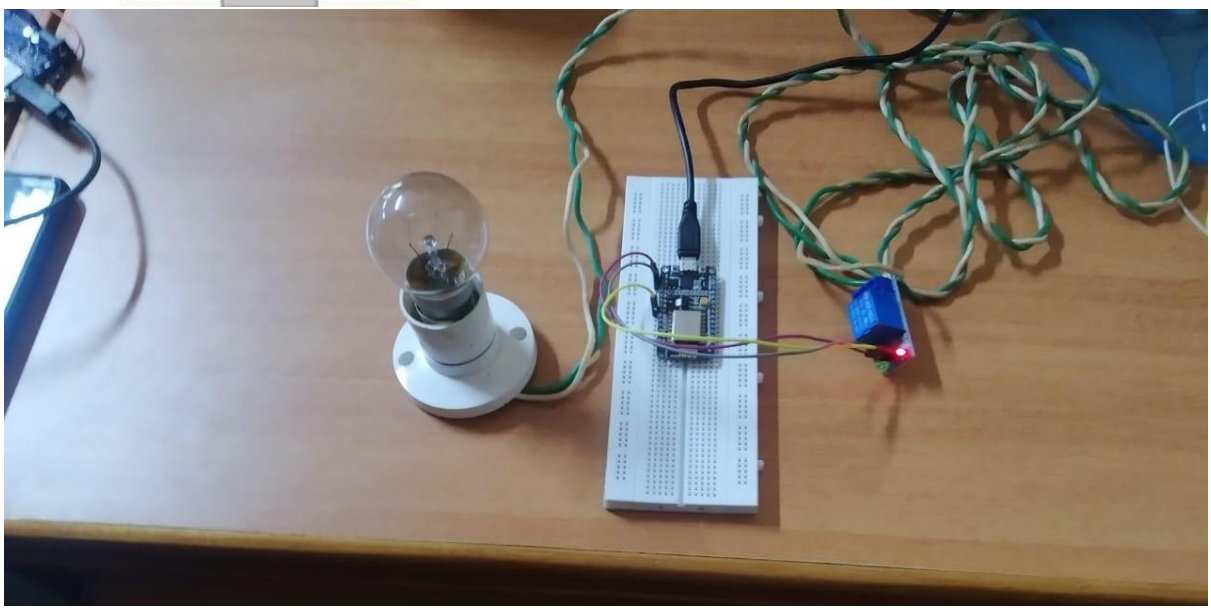
S. No	Item	No's
1.	RELAY(HW307)	1
2.	NODEMCU ESP8266	1
3.	JUMPER WIRES	10
4.	ARDUINO IDE SOFTWARE	-
5.	230V BULB	1
6.	BULB HOLDER	1
7.	USB CABLE	1
8.	BREAD BOARD	1
9.	ELECTRICAL WIRE	3 METRE.

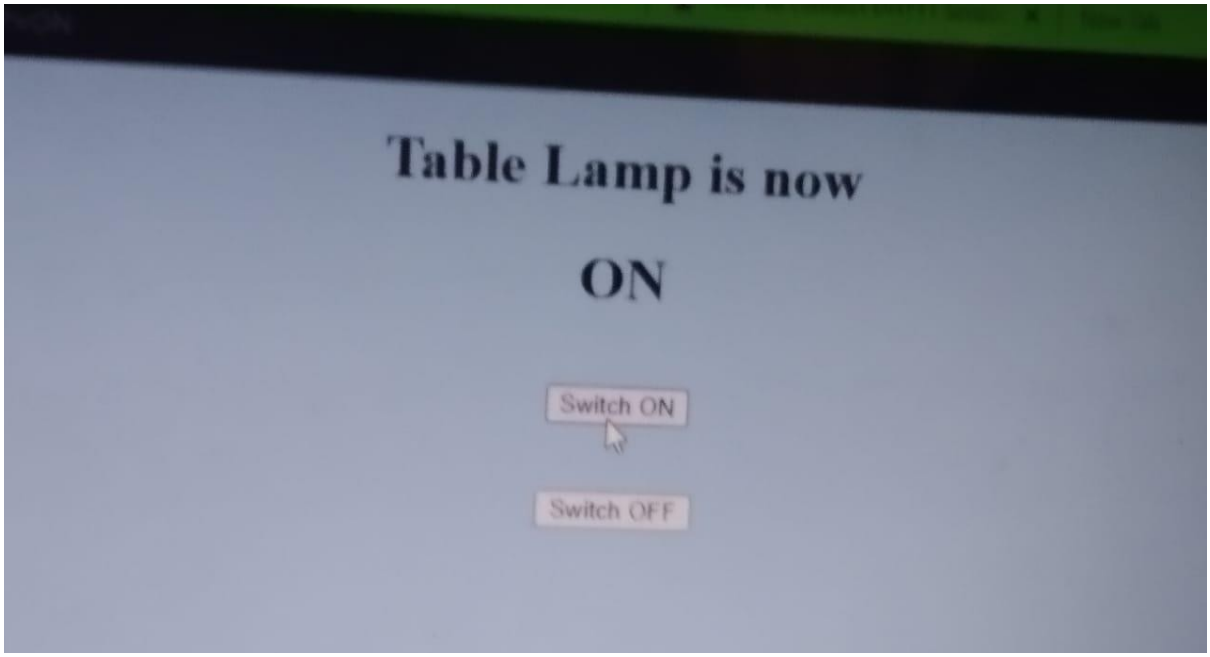
Block Diagram:



Implementation:







Step 1:SETUP AND CONNECTION.

Connect the Relay With NODEMCU

1. Connect 3V3 Pin from NODEMCU to Vcc Pin of Relay.

2. Connect GND Pin from NODEMCU to GND pin of Relay.

3. Connect D5 Pin from NODEMCU to IN Pin of Relay.

Connect Relay To AC Power Supply.

1.Connect Negative Terminal of AC Supply to input Junction of Relay.

2.Connect Output Junction From Relay to Neutral Junction of BULB.

3.Connect Phase of Bulb to AC Supply.

Step 2:PROGRAM

```
#include <ESP8266WiFi.h>


const char* ssid = "comfiny";

const char* password = "welcome2018";


int relay_pin = 14;

WiFiServer server(80);
```

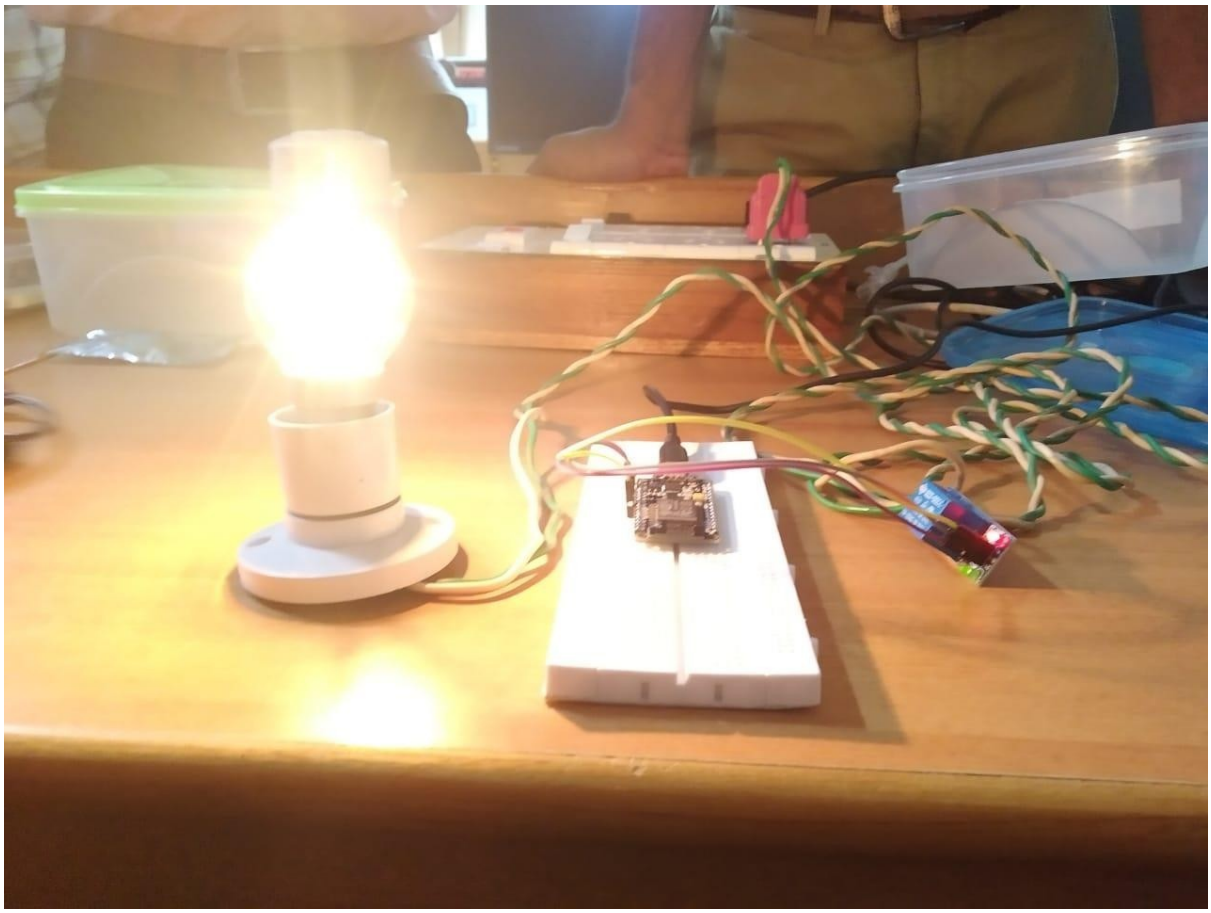
```
void setup()
{
  Serial.begin(115200);
  pinMode(relay_pin, OUTPUT);
  digitalWrite(relay_pin, HIGH);

  Serial.print("Connecting to the WIFI Network\n");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(" * ");
  }
  Serial.println("\nSuccessfully connected to "+String(ssid));
  server.begin();
  Serial.println("\nServer is Active");
  Serial.println("\nType below mentioned IP address in your browser\n");
  Serial.println(WiFi.localIP());
}

void loop()
{
  WiFiClient client = server.available();
  if (!client)
  {
    return;
  }
  while(!client.available())
  {
    delay(1);
  }
  String request = client.readStringUntil('\r');
  client.flush();
```

```
 int value = HIGH;
 if (request.indexOf("/LAMP=ON") != -1)
 {
  digitalWrite(relay_pin, LOW);
  value = LOW;
 }
 if (request.indexOf("/LAMP=OFF") != -1)
 {
  digitalWrite(relay_pin, HIGH);
  value = HIGH;
 }
 client.println("HTTP/1.1 200 OK");
 client.println("Content-Type: text/html");
 client.println(""); client.println("<!
 DOCTYPE HTML>");
 client.println("<html>");
 client.println("<style> body {background-color:#B2D1D5;}</style>");
 client.print("<center><h1>Table Lamp is now<h1></center>");
 if(value == LOW)
 {
  client.print("<center><h1><b>ON</b></h1></center>");
 }
 else
 {
  client.print("<center><h1><b>OFF</b></h1></center>");
 }
 client.println("<br><a href=\"/LAMP=ON\"\"><center><button>Switch ON
</button></center></a><br><br>");
 client.println("<a href=\"/LAMP=OFF\"\"><center><button>Switch OFF
</button></center></a><br/>");
 client.println("</html>");
}
```

Step 3:Upload and Run

Connect Arduino to the computer and upload the program.Now Go to given IP ADDRESS Using Browser to View the Webpage .Using Webpage the Bulb can be Controlled(TURN ON/OFF).

OUTPUT:



RESULT:

Thus, to design and control a bulb via webpage using NODEMCU ESP8266 was Performed Successfully.

Exp no: 9 PUSH IOT SENSOR DATA FOR CLOUD STORAGE AND APPLY SIMPLE DATA ANALYTICS

Date:

AIM:

To push IoT sensor data for cloud storage and apply simple data analytics.

COMPONENTS:

SL.NO	ITEM	NO'S
1	ThingSpeak	-

IMPLEMENATION:

Step 1:

To work with ThingSpeak, a free account can be created. After successfully signing up on ThingSpeak, a channel is created on the dashboard and a 'Write Key' is generated, which is to be used in the Arduino code.

Step 2:

The source code that can be customised and executed on Arduino IDE so that real-time data can be fetched on board and then transmitted to ThingSpeak. For simplicity, the execution of the code is done for NodeMCU hardware, which has inbuilt Wi-Fi for sending the data to the IoT platform.

Step 3:

```
#include <ESP8266WiFi.h>

const char* mynetwork = "<Wi-Fi Network Name>";
```

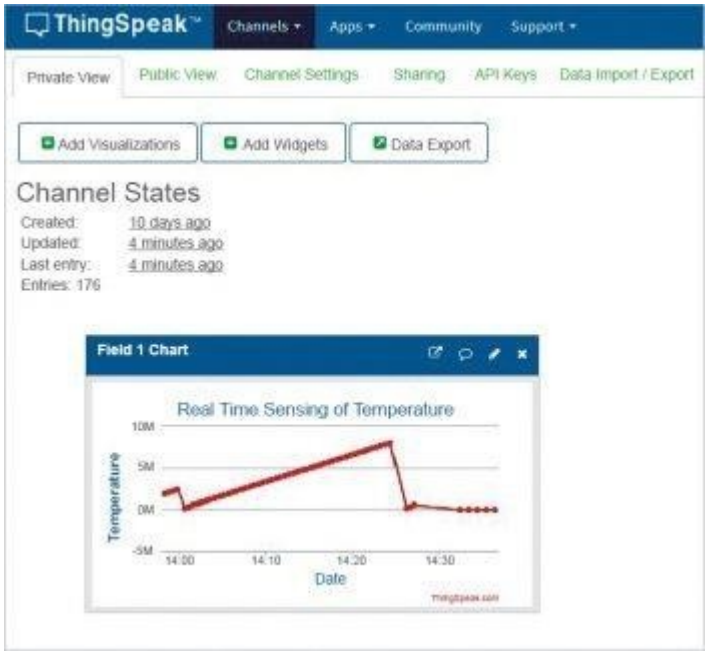
```
const char* mypassword = “<Wi-Fi Network Mypassword>”;
const char* thingspeakhost = “api.thingspeak.com”;
char tsaddress[] = “api.thingspeak.com”;
String ThingSpeakAPI = “<ThingSpeak Write Key>”;
const int UpdateInterval = 30 * 1000;
long ConnectionTime = 0;
boolean connectedlast = false;
int fcounter = 0;
float value = 100;
WiFiClient client;
void setup()
{
Serial.begin(9600);
startEthernet();
}
void loop()
{
String RecordedAnalogInput0 = String(value++, DEC);
if(client.available())
{
char c = client.read();
}
if(!client.connected() && connectedlast)
{
client.stop();
```

```
}
if(!client.connected() && (millis() - ConnectionTime > UpdateInterval))
{
updateThingSpeak("field1="+RecordedAnalogInput0);
}
if(fcounter > 3) {startEthernet();}
connectedlast = client.connected();
}
void updateThingSpeak(String tsData)
{
if(client.connect(tsaddress, 80))
{
client.print("POST /updateHTTP/1.1\n");
client.print("Thingspeakhost: api.thingspeak.com\n");
client.print("Connection: close\n");
client.print("X-THINGSPEAKAPIKEY: "+ThingSpeakAPI+"\n");
client.print("Content-Type: application/x-www-form-urlencoded\n");
client.print("Content-Length: ");
client.print(tsData.length());
client.print("\n\n");
client.print(tsData);
ConnectionTime = millis();
if(client.connected())
{
Serial.println("Connecting to ThingSpeak...");
```

```
Serial.println();
fcounter = 0;
}
else
{
fcounter++;
Serial.println("Connection to ThingSpeak failed ("+String(fcounter, DEC)+")");
Serial.println();
}
}
else
{
fcounter++;
Serial.println("Connection to ThingSpeak Failed ("+String(fcounter, DEC)+")");
Serial.println();
ConnectionTime = millis();
}
}
void startEthernet()
{
fcounter = 0;
client.stop();
Serial.println("Connecting");
WiFi.begin(mynetwork, mypassword);
while(WiFi.status() != WL_CONNECTED) {
```

delay(500); } }

Output:



RESULT:

Thus, IoT sensor data is pushed to cloud storage and simple data analytics has been applied successfully.